AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for executing a fail instruction to facilitate transactional execution on a processor, comprising:

~~executing a start transactional execution instruction to start transactionally executing a block of instructions within a program, wherein the start transactional execution instruction marks the beginning of a block of instructions to be executed transactionally, and wherein the start transactional execution instruction specifies an action to take if transactional execution of the block of instructions fails, and wherein the action to take can include acquiring a lock on the block of instructions;~~

transactionally executing a block of instructions within a program;

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes; and

if the fail instruction is encountered during the transactional execution, terminating the transactional execution without committing results of the transactional execution to the architectural state of the processor, wherein terminating the transactional execution involves branching to a location specified by the fail instruction or to a location specified by a start transactional execution (STE) instruction at the beginning of the transactional execution; or setting state information in the processor and continuing the transactional execution, wherein the processor handles the failure later.

~~retrying the transactional execution a specified number of times, and~~

~~if the fail instruction continues to be encountered after the specified number of times, obtaining a lock for the block of instructions.~~

2. (Original) The method of claim 1, wherein terminating the transactional execution involves discarding changes made during the transactional execution.

3. (Original) The method of claim 2, wherein discarding changes made during the transactional execution involves:

discarding register file changes made during the transactional execution;

clearing load marks from cache lines;

draining store buffer entries generated during transactional execution; and

clearing store marks from cache lines.

4-5. (Cancelled)

6. (Original) The method of claim 1, wherein terminating the transactional execution additionally involves attempting to re-execute the block of instructions.

7. (Previously Presented) The method of claim 1, wherein if the transactional execution of the block of instructions is successfully completed, the method further comprises:

atomically committing changes made during the transactional execution; and

resuming normal non-transactional execution.

8. (Original) The method of claim 1, wherein potentially interfering data accesses from other processes are allowed to proceed during the transactional execution of the block of instructions.

9. (Original) The method of claim 1, wherein if an interfering data access from another process is encountered during the transactional execution, the method further comprises:

discarding changes made during the transactional execution; and

attempting to re-execute the block of instructions.

10. (Original) The method of claim 1, wherein the block of instructions to be executed transactionally comprises a critical section.

11. (Original) The method of claim 1, wherein the fail instruction is a native machine code instruction of the processor.

12. (Original) The method of claim 1, wherein the fail instruction is defined in a platform-independent programming language.

13. (Currently Amended) A computer system that supports a fail instruction to facilitate transactional execution, comprising:

a processor; and

an execution mechanism within the processor;

~~wherein the execution mechanism is configured to execute a start transactional execution instruction to transactionally execute a block of instructions within a program, wherein the start transactional execution instruction marks the beginning of a block of instructions to be executed transactionally, and wherein the start transactional execution instruction specifies an action to take if~~

~~transactional execution of the block of instructions fails, and wherein the action to take can include acquiring a lock on the block of instructions;~~

wherein the execution mechanism is configured to transactionally execute a block of instructions within a program;

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes; and

wherein if the fail instruction is encountered during the transactional execution, the execution mechanism is configured to:

terminate the transactional execution without committing results of the transactional execution to the architectural state of the processor, wherein terminating the transactional execution involves branching to a location specified by the fail instruction or to a location specified by a start transactional execution (STE) instruction at the beginning of the transactional execution, or to set state information in the processor and continue transactional execution, wherein the execution mechanism is configured to handle the failure later

~~retry the transactional execution a specified number of times, and~~

~~if the fail instruction continues to be encountered after the specified number of times, obtain a lock for the block of instructions.~~

14. (Original) The computer system of claim 13, wherein while terminating the transactional execution, the execution mechanism is configured to discard changes made during the transactional execution.

15. (Original) The computer system of claim 14, wherein while discarding changes made during the transactional execution, the execution mechanism is configured to:

discard register file changes made during the transactional execution;

clear load marks from cache lines;

drain store buffer entries generated during transactional execution; and to

clear store marks from cache lines.

16-17. (Cancelled)

18. (Original) The computer system of claim 13, wherein while terminating the transactional execution, the execution mechanism is additionally configured to attempt to re-execute the block of instructions.

19. (Previously Presented) The computer system of claim 13, wherein if the transactional execution of the block of instructions is successfully completed, the execution mechanism is configured to:

atomically commit changes made during the transactional execution; and to

resume normal non-transactional execution.

20. (Original) The computer system of claim 13, wherein the computer system is configured to allow potentially interfering data accesses from other processes to proceed during the transactional execution of the block of instructions.

21. (Original) The computer system of claim 13, wherein if an interfering data access from another process is encountered during the transactional execution, the execution mechanism is configured to:

discard changes made during the transactional execution; and to

attempt to re-execute the block of instructions.

22. (Original) The computer system of claim 13, wherein the block of instructions to be executed transactionally comprises a critical section.

23. (Original) The computer system of claim 13, wherein the fail instruction is a native machine code instruction of the processor.

24. (Original) The computer system of claim 13, wherein the fail instruction is defined in a platform-independent programming language.

25. (Currently Amended) A computing means that supports ~~that supports~~ a fail instruction to facilitate transactional execution, comprising:

a processing means; and

an execution means within the processing means;

~~wherein the execution means is configured to execute a start transactional execution instruction to transactionally execute a block of instructions within a program, wherein the start transactional execution instruction marks the beginning of a block of instructions to be executed transactionally, and wherein the start transactional execution instruction specifies an action to take if transactional execution of the block of instructions fails, and wherein the action to take can include acquiring a lock on the block of instructions;~~

wherein the execution means is configured to transactionally execute a block of instructions within a program;

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes; and

wherein if the fail instruction is encountered during the transactional execution, the execution means is configured to:

terminate the transactional execution without committing results of the transactional execution to the architectural state of the processor, wherein terminating the transactional execution involves branching to a location specified by the fail instruction or to a location specified by a start transactional execution (STE) instruction at the beginning of the transactional execution, or to set state information in the processor and continue transactional execution, wherein the execution means is configured to handle the failure later.

~~retry the transactional execution a specified number of times, and~~

~~if the fail instruction continues to be encountered after the specified number of times, obtain a lock for the block of instructions.~~